

## 智能设备产品接入Swaiot开放平台介绍

以下以WIFI智能灯产品为案例，说明如何完成一个传统设备的Swaiot物端接入基本流程：

### 简介：

通过使用STM32来实现一款十分简单的物联网灯。在这个过程中需要使用到：SWAIOT对接工具（iot-tools）、MDK5。通过三个独立按键加一只LED实现本地功能（实际产品中往往使用组合键或长按触发等操作，当前为了描述简单，分别使用了三个独立按键）。

SWAIOT对接工具工具下载地址：

[https://github.com/swaiot/swaiot\\_tools](https://github.com/swaiot/swaiot_tools)

DEMO下载地址：

[https://github.com/swaiot/swaiot\\_tools-demo](https://github.com/swaiot/swaiot_tools-demo)

本说明版权属于 深圳创维-RGB电子有限公司，由SWAIOT实验室维护。如有更改，恕不另行通知。  
20206.16

---

### 开发说明：

#### 准备工作

- 1、在SWAIOT开放平台或者对接工程师手中申请报备设备三元组数据（设备类型，设备型号，厂家id），并设计通讯协议表。
- 2、新建STM32F103的MDK工程。实现LED开关功能。
- 3、在此工程中添加三个中断按键（LED开关、配网、工厂自检）。
- 4、在程序中添加串口，设置波特率：9600、数据位：8、停止位：1、校验：none。
- 5、使用“SWAIOT对接工具（iot-tools）”配置并导出对接SDK。配置表如下（工具功能配置表会随着导出sdk时自动生成。）：

品牌Id	1	创维				
品类Id	40	台灯				
型号	"CTZ-1018"	型号				
序号	属性	范围	读写	属性名称	属性类型	属性备注
1	开关	0~1	r/w	"STATUS"	uint8	0: 关; 1: 开
属性设置	支持uint8	支持uint16	支持uint32	支持float	支持string	
状态	TRUE	FALSE	FALSE	FALSE	FALSE	
平台配置						
	平台配置	int(32bit)				
	Endian配置	Little-Endian				
功能配置						
1	支持升级	FALSE				
2	支持智屏扩展	FALSE				
3	监听模组状态	FALSE				
4	支持时间获取	FALSE				
5	支持SSID获取	FALSE				
6	支持RSSI获取	FALSE				
7	支持 IP 获取	FALSE				

品牌Id	1	创维				
8	支持MAC获取	FALSE				
优化设置						
功能	支持循环缓冲	循环缓冲大小	支持延迟发送			
状态	TRUE	128	TRUE			

## 对接SDK

1、将导出的“swaiot”和“swaiot/lib”两个文件夹添加至Include Pahts中。

2、对iot\_user\_config.h文件的修改及调用，适配串口发送宏定义：

①、修改如下代码中的宏定义“SEND\_TO\_NET\_CMD”，将宏定义所替换的字符串替换为的串口发送函数。

```
#define SEND_TO_NET_CMD(dataP,dataSize)    sendStringUSART2(dataP,dataSize)
```

注：函数只接收单个byte。如果存在多个，请按顺序逐次调用并传参。

②、修改MCU接收中断（也可以是任意处理接收数据的代码节点，不局限于中断），添加接收函数。

```
void USART2_IRQHandler(void)
{
    if(USART_GetITStatus(USART2,USART_IT_RXNE))
    {
        // add swaiotPushUartData(data);
        swaiotPushUartData(USART_ReceiveData(USART2));
    }
}
```

③、在主函数中初始化SDK，并定时调用处理函数。

**函数initlotCore：**在主函数中初始化调用initlotCore函数，函数参数：（此函数在本程序中无效）

version： 字符型指针，指向一个长度为4的char形数组。[1.0.0.0]代表版本：1.0.0.0

（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

platformName： 字符型指针，指向一个字符串（以\0结尾），代表平台名称。

（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

serialNumber： 字符型指针，指向一个长度为32的char形数组。默认全为0。

（固件版本号，仅对需要升级或对接智慧屏的产品有效。）

**函数swaiotEventHandler\_10HZ：**请在任意位置以100MS的间隔调用

(100ms只是建议值，实际调用中不要求严格的保持100ms)

```
#include "iot_user_config.h"
.....

uint8_t sn_num[32]={0x00};    // 本程序不使用此功能，但支持智慧屏或升级的程序需要添加
char ver_mcu[4]={1,0,0,0};    // 本程序不使用此功能，但支持智慧屏或升级的程序需要添加
char *type_mcu = "STM32F103"; // 本程序不使用此功能，但支持智慧屏或升级的程序需要添加
.....

int main(void)
{
    .....

    initIotCore(ver_mcu,type_mcu,sn_num); // 本程序不使用此功能，但支持智慧屏或升级的程
序需要添加
    uploadInitialStatus();
    .....

    while (1){
        checkLedStatus();           // check and update status, 属性上报检测
        checkControlBtn();          // check and update status, 配网、工厂按键检测
        if(frame_10Hz ==1){         // 10Hz flag
            frame_10Hz = 0;          // clean flag
            swaiotEventHandler_10HZ();// 周期调用IOT处理函数
        }
    }
}
```

#### ④、添加模块功能（配网、工厂测试）：

在按键配网和工厂测试回调函数中添加“配网”、“工厂测试”两个功能。

```
void checkControlBtn(void)
{
    if(connectFlg == 1)
    {
        connectFlg = 0;
        sendCmdSetModuleConnect(); // 工具自动生成的配网函数
    }
    if(facTestFlg == 1)
    {
        facTestFlg = 0;
        sendCmdSetModuleFacTest(); // 工具自动生成的工厂测试函数
    }
}
```

#### ⑤、属性发送函数

将按键LED开关回调函数中添加自动生成的Led控制函数。此函数的函数名称是根据属性表中属性名称自动生成的（属性名称："STATUS"）

```

void checkLedStatus(void)
{
    if(keyFlag == 1){
        keyFlag = 0;
        if(GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_5) == Bit_RESET){
            sendcmdPropStatus(1); // 工具自动生成的状态更新函数
        }else{
            sendcmdPropStatus(0); // 工具自动生成的状态更新函数
        }
    }
}

```

3、对iot\_receive\_handler.c文件进行修改。

①、修改数据更新回调函数,添加LED状态控制代码。

```

void receiveAProps8(char *name, iot_s8_t value){ // 此函数由工具自动生成
// has a s8 data
if(0 == strcmp(name, PROD_STATUS_NAME)){ // 判断属性名称, 工具自动生成
    if(value == 0){
        GPIO_SetBits(GPIOB, GPIO_Pin_5); // 更新LED状态
        sendcmdPropStatus(0);
    }else{
        GPIO_ResetBits(GPIOB, GPIO_Pin_5);
        sendcmdPropStatus(1);
    }
}
}

```

②、可选的其他回调功能。

如果想通过模块状态显示wifi连接状态, 在如下回调中监听模块状态。(须在工具中开启监听模组状态)

```

/*****
Function....: moduleStateChangeHandler
Description.: Callback function for updating module state
Parameters..: ENUM_MODULE_STATE_SKY state : it is an enumeration of states.
Please refer to the iot_base.h
Return.....: NONE
*****/
void moduleStateChangeHandler(ENUM_MODULE_STATE_SKY state){
    // update state
}

```

如果配置了其他扩展功能, 请根据代码注释添加相应的回调。

备注: 为了保证C51与其他平台的代码兼容性, 并未使用指针函数。所有回调接口需要在iot\_receive\_handler.c文件中实现。